

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

**THIS PAGE BLANK (USPTO)**

JAPANESE LAID-OPEN PATENT APPLICATION  
2000-148491(P2000-148491A)

(19) Japan Patent Office (JP) (11) Publication No. 2000-148491  
(12) Published Unexamined Patent Application (A)  
(43) Publication Date May 30, 2000  
(51) Int. Cl.7 Identification Code FI  
G 06 F 9/38 350 G06F 9/38 350A  
310 310F  
370 370C  
Examination request: done Number of claims 18 OL (totally 21 pages)

---

(54) Title of the Invention

SYSTEM AND METHOD FOR HANDLING LOAD AND/OR STORE OPERATIONS IN  
A SUPERSCALAR

(21) Application No. 2000-8268(P2000-8268)  
(62) Division of Application JP Appl. No. H6-509063  
(22) Date of Filing September 29, 1993 (Heisei 5)  
(33) Priority Claim 07/954,084  
(32) Priority Date September 29, 1992 (Heisei 4)  
(33) Priority Country The United States  
(71) Applicant Seiko Epson Corporation  
4-1 Nishishinjuku 2-chome Shinjuku-ku Tokyo  
(72) Inventor WANG, Johannes  
25 King Street, Redwood City, CA 94062 (US)  
(74) Agent Masahiro HIRUKAWA, Attorney and 7 others

[CLAIMS]

[Claim 1] A microcomputer system to execute program streams, said microcomputer system comprises;

- (a) an instruction fetch unit to fetch instructions from a memory system and provide predetermined plural number of instruction to an instruction buffer,
- (b) an execution unit to execute said plural of said instruction from said instruction buffer out of order, wherein said execution unit includes a

**THIS PAGE BLANK (USPTO)**

load store unit set up to request the memory system to load all instructions in said instruction buffer out of order, and to request to store all instructions in said instruction store in order, said load store unit includes;

- (i) an address conflict mean to detect address confliction or writing pending between said plural of said instruction respectively and to notify with signals, when address confliction and writing pending are not detected, said load store unit request said loading,
- (ii) an address path set up to control plural of address related to said plural of said instruction corresponding to said address conflict mean, and to provide addresses to said memory system,
- (iii) a data path to transmit load and/or store data to said memory system or said execution unit, or from said memory system or said execution unit, said data path is set up to align data returned from said memory system, and to return data on border of words from said memory system to said execution unit aligned correctly.

[Claim 2] In the microcomputer system of Claim 1, said address path includes plural of address buffer to store high order byte and low order byte of said load and/or said store request.

[Claim 3] In the microcomputer system of Claim 1, said load store unit further includes a mean to request multiple-memory to the memory when said data is on the border of words.

[Claim 4] In the microcomputer system of Claim 1, a data address function unit set up to calculate addresses for said plural of instruction, and to provide said calculated addresses to said load store unit is also included.

[Claim 5] In the microcomputer system of Claim 4, a virtual memory unit set up to provide physical address translation created by a virtual address to said execution unit and said load store unit is also included.

[Claim 6] In the microcomputer system of Claim 5, said load store unit must have physical address from said data address function unit and said virtual memory unit before memory request is allowed.

**THIS PAGE BLANK (USPTO)**

[Claim 7] In the microcomputer system of Claim 1, wherein said instruction is a CISC instruction, and said instruction execution unit further includes a decoding mean to decode said CISC instruction to a RISC instruction.

[Claim 8] In the microcomputer system of Claim 1, said load store unit further includes a merging mean to merge data received from the memory with initial contents of a destination register.

[Claim 9] In the microcomputer system of Claim 1, a mean to transmit load and/or execution data from said execution unit to said data path directly so that follow-on store operation can be performed immediately.

[Claim 10] In the microcomputer system of Claim 1, a history pointer to indicate relative age of instruction in said instruction buffer is also included.

[Claim 11] In the microcomputer system of Claim 1, said conflict mean indicates load dependency by examining whether address conflict or pending store address exist.

[Claim 12] In the microcomputer system of Claim 1, a mean to prevent load bypassing of load instruction which changes state of the microcomputer system falsely is included.

[Claim 13] In the microcomputer system of Claim 1, said execution unit has plural of function unit, the microcomputer system includes a mean to monitor said plural of function unit to provide store data to said load/store data path directly.

[Claim 14] In the microcomputer system of Claim 1, said load store unit has independent load/store data for floating point arithmetic.

[Claim 15] In the microcomputer system of Claim 1, said execution unit includes a register file including plural of real register and plural of temporary register.

**THIS PAGE BLANK (USPTO)**



[Claim 16] In the microcomputer system of Claim 1, each instruction buffer includes plural of bucket, and each said bucket includes load and store, only load or only store to the same address or does not include load and store respectively.

[Claim 17] In the microcomputer system of Claim 1, said load store unit gives priority higher than said load request to said store request.

[Claim 18] In the microcomputer system of Claim 1, a mean to prevent load bypass of load which is impossible to cash is included.

**THIS PAGE BLANK (USPTO)**

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-148491

(P2000-148491A)

(43) 公開日 平成12年5月30日(2000.5.30)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 9/38	3 5 0	G 0 6 F 9/38	3 5 0 A
	3 1 0		3 1 0 F
	3 7 0		3 7 0 C

審査請求 有 請求項の数18 O L (全 21 頁)

(21) 出願番号 特願2000-8268(P2000-8268)  
(62) 分割の表示 特願平6-509063の分割  
(22) 出願日 平成5年9月3日(1993.9.3)  
(31) 優先権主張番号 07/954, 084  
(32) 優先日 平成4年9月29日(1992.9.29)  
(33) 優先権主張国 米国 (U S)

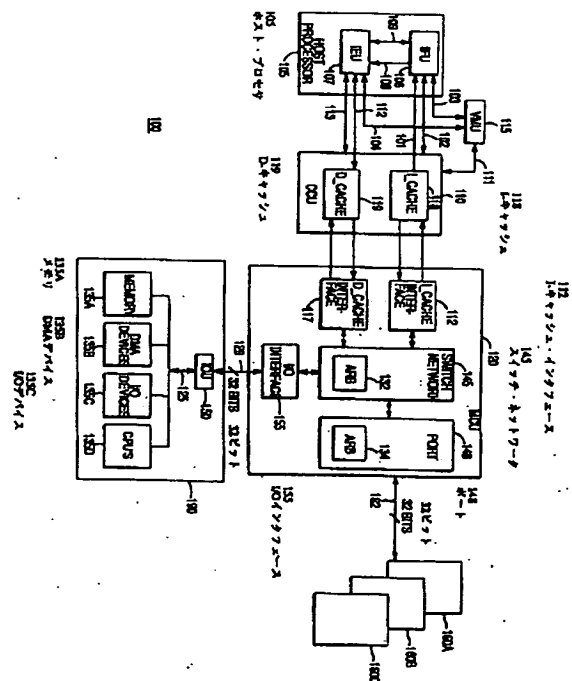
(71) 出願人 000002369  
セイコーエプソン株式会社  
東京都新宿区西新宿2丁目4番1号  
(72) 発明者 センター シュリル  
アメリカ合衆国 カリフォルニア州  
95014 クバチーノ, セレステ サークル  
20746  
(72) 発明者 ワング ジョハネス  
アメリカ合衆国 カリフォルニア州  
94062 レッドウッド シティ, キングス  
トリート 25  
(74) 代理人 100092495  
弁理士 蛭川 昌信 (外7名)

(54) 【発明の名称】 マイクロコンピュータシステム

(57) 【要約】

【課題】 依存性によるデータ・エラーの発生を回避し、複数の命令を平行処理的にアウト・オブ・オーダーで実行可能にする。

【解決手段】 メモリシステム(110)から命令をフェッチする命令フェッチ・ユニットと、アウト・オブ・オーダーでメモリシステムにロード要求を行い、イン・オーダーでストア要求を行うように適応させたロードストア・ユニットを含む実行ユニット(107)とを含み、ロードストア・ユニットがアドレス衝突手段と、アドレスバスと、メモリシステムから返されたデータを整理して、ワード境界上にあるデータをメモリシステムから実行ユニットへ正しく整理して返せるように構成されているデータベースを含むマイクロコンピュータシステムである。



(2)

## 【特許請求の範囲】

【請求項1】 プログラムストリームを実行するためのマイクロコンピュータシステムであって、前記マイクロコンピュータシステムが、

(a) メモリシステムから命令をフェッチし、所定の複数の命令を命令バッファへ提供するための命令フェッチ・ユニットと、

(b) 前記命令バッファからの前記複数の前記命令をアウト・オブ・オーダーで実行するための実行ユニットであって、前記命令バッファ内のすべての命令に関しては、  
アウト・オブ・オーダーでメモリシステムにロード要求を行い、前記命令ストア内のすべての命令に関しては、イン・オーダーでストア要求を行うように適応させたロードストア・ユニットを含む実行ユニットとを含み、前記ロードストア・ユニットが

(i) 前記複数の前記命令のそれぞれの間にアドレス衝突および書き込みペンディングが存在するかどうかを検出して信号で知らせるためのアドレス衝突手段であって、アドレス衝突および書き込みペンディングが検出されないときは、前記ロードストア・ユニットは前記ロード要求を行うアドレス衝突手段と、

(i i) 前記アドレス衝突手段に対応して、前記複数の前記命令に関連ある複数のアドレスを管理するよう、また前記メモリシステムへアドレスを提供するよう適応させたアドレスバスと、

(i i i) 前記メモリシステムおよび前記実行ユニットへ、また前記メモリシステムおよび前記実行ユニットから、ロードおよび／またはストアデータを転送するためのデータバスであって、前記メモリシステムから返されたデータを整列して、ワード境界上にあるデータを前記メモリシステムから前記実行ユニットへ正しく整列して返せるように構成されているデータバスとを含むマイクロコンピュータシステム。

【請求項2】 前記アドレスバスが、前記ロードおよび／または前記ストア要求の上位バイトおよび下位バイトを記憶するための複数のアドレスバッファを含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項3】 前記ロードストア・ユニットが、前記データがワードの境界上にあるときは、メモリに対して多重メモリ要求を行うための手段をさらに含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項4】 さらに、前記複数の命令のためにアドレスを計算するように、また前記計算されたアドレスを前記ロードストア・ユニットに提供するように適応させたデータアドレス機能ユニットを含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項5】 仮想アドレスから生成された物理アドレス翻訳を前記実行ユニットおよび前記ロードストア・ユニットに提供するよう適応させた仮想メモリユニットを

2

含むことを特徴とする請求項4記載のマイクロコンピュータシステム。

【請求項6】 前記ロードストア・ユニットが、メモリ要求を行うことができる前に前記データアドレス機能ユニットおよび前記仮想メモリユニットからの物理アドレスを持たなくてはならないことを特徴とする請求項5記載のマイクロコンピュータシステム。

【請求項7】 前記命令がCISC命令であり、前記命令実行ユニットがさらに、前記CISC命令をRISC命令へ復号するための復号手段を含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項8】 前記ロードストア・ユニットが、さらにメモリから受けたデータを宛先レジスタの当初の内容と併合させる手段を含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項9】 ロードおよび／または実行データを、前記実行ユニットから前記データバスへ直接転送して、後続のストア演算をその後直ちに行えるようにする手段を、さらに含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項10】 さらに、前記命令バッファ内の命令の相対的年齢を示す履歴ポインタをさらに含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項11】 アドレス衝突またはペンディングストアアドレスがあるかどうかを調べることによって、前記衝突手段がロード依存を示すことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項12】 さらに、マイクロコンピュータシステムの状態を不正に変えるようなロード命令のロードパイパスを防止するための手段を含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項13】 前記実行ユニットが複数の機能ユニットを有し、ストアデータを前記ロード／ストアデータバスへと直接提供するために、マイクロコンピュータシステムが前記複数の機能ユニットの結果を監視する手段を含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項14】 前記ロードストア・ユニットが、浮動小数点演算のための別個のロード／ストアデータを有していることを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項15】 前記実行ユニットが、複数の実レジスタと複数の一時レジスタを含むレジスタファイルを含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項16】 各命令バッファが複数のバケットを含み、前記バケットのそれぞれが、同じアドレスに対してロードとストアか、ロードだけか、ストアだけかを含み、またはロードとストアを含まないことを特徴とする請求項1記載のマイクロコンピュータシステム。

(3)

3

【請求項17】 前記ロードストア・ユニットが、前記ロード要求を超える優先権を前記ストア要求に与えることを特徴とする請求項1記載のマイクロコンピュータシステム。

【請求項18】 さらに、キャッシュ不可能なロードのロードバイパスを防止するための手段を含むことを特徴とする請求項1記載のマイクロコンピュータシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は一般的にはスーパースカラ・マイクロプロセッサの設計に関し、より具体的には、命令をアウト・オブ・オーダーで実行するマイクロプロセッサに於けるロード及びストア動作を扱うシステム並びに方法に関する。本出願は、本出願の代理人に譲渡されている下記の出願に関連するものである。すなわち、ニューエン (Nguyen) その他による1991年7月8日出願の米国特許出願番号 07/727,058 (代理人整理番号SP021)、「拡張可能RISCマイクロプロセッサ・アーキテクチャ」(EXTENSIBLE RISC MICROPROCESSOR ARCHITECTURE)、及び'058 出願の継続出願である、1992年1月8日出願の出願番号07/817,809に関連する。上記出願の開示を参照することによって当該特許出願の明細書の記載内容が本明細書に組み込まれているものとする。

【0002】

【従来の技術】 スーパースカラ縮小命令セット・コンピュータ (RISC) に於ける大きな課題は如何にして斯かる命令実行の本質的な問題である依存性によるデータ・エラーの発生を回避しつつ、複数の命令を平行処理的に、アウト・オブ・オーダーで実行できるか、という点に関する。RISCプロセッサに於ける最も簡単な命令発行方針は命令をプログラムでの順序と同じ順序で発行し (イン・オーダー発行)、結果をまた同じ順序で書き出すことである (イン・オーダー完了)。アウト・オブ・オーダー完了はイン・オーダー完了に比して複雑であるが、同じ種類の動作に対してスーパースカラ・プロセッサの性能向上に効果がある。例えば、アウト・オブ・オーダー完了はロード又は浮動小数点演算等の長い待ち時間動作の性能改善のために使用される。機能ユニット内で実行中の命令の数に制限はないが、その最大数は全ての機能ユニット内のパイプライン段階の数である。この場合、命令はアウト・オブ・オーダーで完了でき、その理由は結果の計算にひとつの機能ユニットが1サイクル以上のサイクルを費やした場合でも命令の発行に停止がないからである。従って、後続の命令が終了した後、機能ユニットが一つの命令を完了しても構わない。

【0003】 下記のコードシーケンスを考えて見る。ただし"op" はオペレーション、"Rn" は番号つきレジスタ、": =" は代入を表わす。

【0004】 R3:=R3 op R5 (1)

R4:=R3 + 1 (2)

4

R3:=R5 + 1 (3)

R7:=R3 op R4 (4)

この場合、一般的にアウト・オブ・オーダーの命令了が可能であっても、第1命令の代入は第3命令の代入の後には完了できない。第1命令と第3命令がアウト・オブ・オーダーで実行された場合、異常且つ不正な値がR3レジスタに残り、例えば第4命令が不正なオペランド値を受け取るような事態が生じる。第3命令の結果は第1命令に対して「出力依存性」を有し、このコードシーケンスに於いて正しい出力値を得るためには第3命令は第1命令の後に完了しなければならない。従って、第3命令の結果が、計算にもっと時間の掛かるより古い命令によって上書きされる場合、第3命令の発行は待たなければならない。

【0005】 アウト・オブ・オーダーの完了は性能の向上をもたらすものであるが、より多くのハードウェア、つまりデータ依存性論理、を必要とする。アウト・オブ・オーダーの完了の場合、データ依存性論理は複雑になる。その理由は、この論理ではデコードされた命令と、全てのパイプライン段階の全ての命令の間に於けるデータ依存性の検査が行なわれるからである。結果が正しい順序で書き出されることを保証するのもハードウェアの役割である。これに対して、イン・オーダー実行に於いては、データ依存性論理はデコードされた命令と、現在実行中の幾つかの命令の間に於けるデータ依存性のみを検査すればよいから、結果は当然正しい順序で書き出される。アウト・オブ・オーダー完了に於いてはまた、機能ユニットは結果バス及びレジスタ・ファイル書き込みポート間での調停を行なわなければならない。その理由は、同時に完了する全ての命令の必要を満たすために十分な数のバス及びポートが存在しないであろうからである。

【0006】 更に、アウト・オブ・オーダーの完了では命令例外の処理がより困難になる。ある条件下で命令が例外を生成した場合、その命令はハードウェアだけでは正しく実行できない。デコードされた命令が資源競合を生じた場合、真の依存性を有する場合、あるいは未完了の命令に対して出力依存性を有する場合、イン・オーダー命令発行プロセッサは命令のデコードを中止する。従って、後続する一つ又は複数の命令が実行可能であっても、プロセッサは競合又は依存性を生じた命令の後に来るものを先読み (lookahead) 処理できない。従来の解決策はデコードを実行段階から分離して、命令が直ちに実行可能であるか否かにかかわらず命令のデコードを継続して行なえるようにすることである。この分離はデコード段階と命令段階の間に「命令ウィンドウ」と呼ばれるバッファを配置することによって実施される。先読みのために、プロセッサは命令をデコードし、ウィンドウ中に場所がある限りデコードされた命令を命令ウィンドウに入れ、それと同時に、実行可能な命令 (すなわち、資源競合又は依存性を持たない命令) を見出すためにウィンド

(4)

5

ウ中の命令を検査する。命令ウィンドウは命令のプールとしての機能があり、この機能によってプロセサは先読みの能力を得る。この能力はウィンドウのサイズとプロセサの命令フェッチ・ユニット (IFU) の性能によって制約されるものである。これによって、各命令の最初のプログラム順序と無関係にウィンドウから発行できるので、命令のアウト・オブ・オーダー発行が可能となる。この場合、命令発行元に対する唯一の制約はプログラムが正常に動作することを保証するのに必要な制約である。

【0007】如何なる特定の命令に於いても、発行に関する制約はイン・オーダー発行の場合と殆ど同じである。すなわち、命令は資源競合又は依存性を持たない場合発行されるのである。アウト・オブ・オーダーの発行によって、プロセサは発行可能なより大きな命令集合を得ることになり、それによって、同時に実行可能な命令をプロセサが見つかる確率が高まる。しかし、命令をアウト・オブ・オーダーで発行する能力によって他の発行制約が生じる。これは命令をアウト・オブ・オーダーで実行する能力に於いて出力依存性の制約が導入されるのに類似している。

【0008】これを理解するために上記のコードシーケンスの例を振り返って見る。第2命令の実行が始まる前には第3命令の代入は完了できない。さもなければ、第3命令が第2命令の第1オペランドを不正に上書きすることがあり得る。第3命令の結果は第2命令の第1入力オペランドに対して「反依存性」を有するといわれる。

「反依存性」という用語は、反依存性制約はそれが逆になった以外には真の依存性に関する制約と同様であることを意味する。第2命令が使用する値を第1命令が生成するかわりに、第2命令は第1命令が使用する値を破壊する値を生成する。これを防止するためには、第2命令が始まるまでプロセサは第3命令を発行してはならない。第2命令は第1命令に依存するから、第3命令は他の点では独立していても第1命令が完了するまで待たなければならない。

【0009】反依存性が重要なのは主に命令がアウト・オブ・オーダーで発行可能な場合である。正常なオペレーション中、停止した命令の入力オペランドは後続する命令によって破壊されることがある。しかし、スカラ・プロセサに於いては、往々にして命令例外は例外条件を修正し、そして問題を生じた命令を再試行することによって処理される。この命令がアウト・オブ・オーダーで完了した場合、その命令の再試行が行なわれた時、後続する命令によってその再試行中の命令の入力オペランドが上書きされることがあり得る。この問題は正確な割り込みをサポートするプロセサでは起こり得ない。この問題の解決には、再起動を可能にするためにプロセサが命令オペランドのコピーを維持する必要があるかもしれない。

【0010】プログラム命令によって行われるオペレーションの代表的な二つのオペレーションはロード及びス

6

トアのオペレーションである。一般的に、ロード及びストア・オペレーションはそれぞれ記憶場所を読み出し、変更する。他のプログラム命令と同様に、ロード及びストアはアウト・オブ・オーダーで実行できる。ロード及びストアは同時にデコード可能であるが、従来的には1サイクル当たり一つのロード又はストアのみが発行される。データ・キャッシュの使用に於いては、通常、ロードはストアに優先される。その理由は多くの場合ロードはプロセサが演算を行うのに必要な値を生成するからである。データ・キャッシュの使用に於いて、ストアがロードと競合する場合、ストアの実行が可能になるまで、通常ストアはストア・バッファに保持される。更に、従来的には、ストアは他のストアに対してプログラム順序で実行され、ロードも含めて全ての他の先行の命令が実行された後にのみ実行される。これによって、データ・キャッシュ使用に於いてのプロセサのイン・オーダー状態が保存される。その理由はキャッシュの更新はそれが絶対に正しく行なわれ得るまで行なわれないからである。ストア・バッファの使用によってストアが正しい順序で保持され、先行の命令が完了するまでストアの完了が延期されるのである。

【0011】ストアは他の先行の命令が実行されるまで保持され、そしてロードはプロセサ中での計算に必要な値を生成するから、ストアに対してロードをプログラム順序に保持することは性能に対して重大な悪影響を及ぼす。全ての先行のストアが完了するまでロードが待たなければならない場合、そしてそのために最も新しいストアに先行する全ての命令が完了するまでロードが待つ場合、ロード・データに依存した、ロードに後続する全ての命令も待つことになる。この性能上の問題を回避するために、ロードはストア・バッファで待機している、先行のストアをバイパスすることができ、ロード・データは後続の計算に於いて使用できる。

【0012】ロードが先行のストアをバイパスすることができる場合、ロードは未だ実行されていない先行のストアからデータを得る必要があるかも知れない。プロセサはロードが先行のストアに対して有する真の依存性を、ロードの仮想記憶アドレスと、全ての未完了の先行ストアの仮想記憶アドレスとを比較することによって検査する (仮想アドレスとは記憶管理ユニットによるアドレス変換が適用される前に、命令によって直接計算されるアドレスである)。ここに於いて、各仮想アドレスに対して一意的なマッピングの存在が仮定される。その理由は2個の異なる仮想アドレスが同じ物理的なアドレスへアクセスすることがないようにするためである。この仮定に基づき、仮想アドレス比較は物理的記憶場所間の全ての依存性を検出する。ロード・アドレスが先行のストアのアドレスと一致する場合、又は先行のストアのアドレスのいずれも未だ計算されていない場合 (この場合、依存性は検出不可能なので、依存性の存在が仮定さ

(5)

7

れる)、一つのロードは一つのストアに対して真の依存性を有する。ロードがストアに依存する場合、データ・キャッシュは正しい値を持たないので、そのロードはデータ・キャッシュによって満たされない。ストアの有効アドレスが後続するロードのアドレスと一致する場合、そのロードはストアが完了するのを待つ代わりに、ストア・データが有効な場合、ストア・バッファによって直接満たされる。

【0013】上述の如く、ロード及びストアは記憶場所に対する反依存性及び出力依存性を回避するような方法で実行される。ロードは先行のストアをバイパスできるが、ストアは先行のロードをバイパスできない。従って、ロードとストアの間には反依存性は存在し得ない。一つのストアは従来他のストアに対して通常プログラム順に発行されるので、ストア間には出力依存性は存在し得ない。

【0014】従来的には、データ・キャッシュに於いてロードは他のロードに対してプログラム順に実行される。当業者の意見ではロードをアウト・オブ・オーダーで実行することにより性能の点で得られる利点はないとのことである。その理由は、古いロードによってプロセッサに供給されるデータは新しいロードによって供給されるデータよりも計算に必要とされることが多いからである。

【0015】上記の概念の詳細は幾つかの刊行物で論じられている。例えば、John L. Hennessyその他著、「Computer Architecture - A Quantitative Approach」(Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990年発行)及びMike Johnson著「Superscalar Microprocessor Design」(Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991年発行)(特に第8章、この章の一部分は上に転載されている)。両書とも参照することによって全文が本明細書に組み込まれているものとする。

【0016】

【発明が解決しようとする課題】本発明はスーパースカラRISC型マイクロプロセッサ・アーキテクチャ環境に於いてメモリからの読み出し及びメモリへの書き込み、或いは入出力に必要なロード及びストア・オペレーションを管理するシステムを提供するものである。

【0017】

【課題を解決するための手段】本発明はプログラム・ストリームを実行するマイクロプロセッサ・システムを提供するもので、このシステムには命令を命令ストアより取り出し、且つ予め決められた複数の命令を命令バッファに供給する命令フェッチ・ユニットが含まれている。更に、命令フェッチ・ユニットと結合している実行ユニットが含まれ、実行ユニットは命令バッファからの複数の命令をアウト・オブ・オーダーで実行するためである。実行ユニットにはロード・ストア・ユニットが含まれ、ユ

8

ニットはアウト・オブ・オーダーなロード要求とイン・オーダーのストア要求をメモリ・システムに対して行なうように適性化されている。従って、本発明のロード/ストア・ユニットの主な目的は、可能な限り、アウト・オブ・オーダーなロード要求を行ない、なるべく速やかにロード・データを命令実行ユニットに返すことである。ロード・オペレーションはアドレス衝突がなく、実行待ちの書き込みオペレーションが存在しない時のみアウト・オブ・オーダーで実行できる。アドレス衝突が発生するのは、古い命令がこれから書き込まれる記憶場所に於いて読み出しが要求された時である。実行待ちの書き込みオペレーションとは、古い命令がストア・オペレーションを要求したがストア・アドレスの計算がまだ行なわれていないことである。データ・キャッシュ・ユニットは位置合わせされていない8バイトのデータを返す。ロード/ストア・ユニットはデータが命令実行ユニット(IEU)に返される前にデータの正しい位置合わせを行う。従って、ロード/ストア・バッファの三つの主要なタスクは(1) アウト・オブ・オーダーのキャッシュ要求の処理、(2) アドレス衝突の検出、及び(3) データの位置合わせである。

【0018】ロード・ストア・ユニットには現在実行中の複数の命令に対応する複数のアドレスを管理するために適性化されたアドレス・バス、現在実行中の複数の命令の各命令間にアドレス衝突及び実行待ちの書き込みオペレーションが存在するかどうかを検出して知らせるアドレス衝突手段が含まれ、そうすることによって、ロード・ストア・ユニットはアドレス衝突も実行待ちの書き込みオペレーションも検出されなかった場合、ロード要求を実行する。ロード・ストア・ユニットは更にデータ・バスで構成され、データ・バスはロード及び1又はストア・データをメモリ・システム及び命令実行ユニットの間で転送する。データ・バスは記憶システムより返されたデータの位置合わせをし、斯くしてキャッシュの4ワード境界と一致しないデータがメモリ・システムから命令実行ユニットに正しいアライメントで返されるようにする。

【0019】

【発明の実施の形態】本発明は添付の特許請求の範囲に具体的に提示されている。本発明の上記の、そして後述の利点の理解を深めるために、次に図面を参照して説明する。すなわち、図1において、本発明の好適な実施例に基づいて、一般的に100で表わされるマイクロプロセッサ・アーキテクチャが配置されている。システム・アーキテクチャ100にはホスト・プロセッサ105、キャッシュ制御ユニット及びメモリ(CCU)110、仮想メモリ・ユニット(VMU)115、入出力サブシステム190、メモリ制御及びインタフェース・ユニット120、及びインタリーブ・オペレーション用に構成されたインタリーブ・メモリ・バンク(160a、160b、160C)(以降、主メモリ

(6)

9

160と称す)が含まれている。主メモリ160は外部データバス162を介してMCU120に接続されている。本発明はマルチプロセサ環境で動作すると予想されるので、その場合、他のプロセサもメモリバス162に接続される。ホストプロセサ105は主メモリ160に於いて各アドレス又は記憶場所に格納されているソフトウェア命令を実行する。これらのソフトウェア命令はホスト・プロセサ105にプログラム・カウンタの制御の下にイン・オーダで転送される。しばしば、命令のうちあるものはホストプロセサ105が一つ又は複数の周辺入出力装置135をアクセスすることを必要とする。

【0020】MCU120は一つの回路であり、この回路によってデータ及び命令はCCU110(D\_\_キャッシュ119とI\_\_キャッシュ118(読み出し専用))、IOU150、及び主メモリ160の間を転送される(読み出されるか書き込まれる)。MCU120にはスイッチ・ネットワーク145が含まれ、それにはスイッチ・アービトレーション・ユニット132、データ・キャッシュ・インタフェース・ユニット117、命令キャッシュ・インタフェース回路112、I/Oインタフェース回路155、及びポートとして知られる一つ又は複数のメモリポート・インタフェース回路148が含まれる。各ポート・インタフェース回路148にはポート・アービトレーション・ユニット134が含まれている。

【0021】スイッチ・ネットワーク145はマスタ装置とスレーブ装置間の通信の手段である。スイッチ・ネットワーク120に対するマスタ装置になり得るのはD\_\_キャッシュ119とI\_\_キャッシュ118、又はI/Oコントローラ・ユニット(IOU)150である。スレーブ装置として機能し得るものは、例えば、メモリ・ポート148又はIOU150である。スイッチ・ネットワーク145の機能はCCU110(I\_\_キャッシュ118及びD\_\_キャッシュ119)とIOU150から様々な命令及びデータ要求を受け取ることである。これらのユニットをバス要求者と呼ぶ。これらの要求を受け取った後、スイッチ・アービトレーション・ユニット132及びポート・アービトレーション・ユニット134は要求を優先度に応じて並べ、適切なメモリポートに渡す(命令アドレスによる)。ポート148、或いは場合によっては複数のポート、は次に必要なタイミング信号を生成し、データを外部バス162に送り、あるいはデータを外部バス162から受け取る。

【0022】命令フェッチ・ユニット(IFU)106及び命令実行ユニット(IEU)107はホスト・プロセサ105の主要な動作構成要素である。IFU106及びIEU107の機能を直接サポートするためにVMU115、CCU110、及びMCU120が配置されている。IFU106の主要な機能は命令の取り出し、IEU107による実行を待つ命令のバッファリング、そして、一般的に、次の命令の取り出しに使用される次の仮想アドレスの計算、である。各命令は命令バス101を介してI\_\_キャッシュ118からIFU106によって同時に取

10

り出される。命令は「バケット」或いは4個の命令の集合に入れられる。命令集合の転送は制御バス102を介して供給される制御信号によって、IFU106及びCCU110の間で調整される。取り出される命令の仮想アドレスはIFU制御及びアドレス・バス103を介してIFU106によってVMU115に供給される。VMU115へのアクセスに関する調停の必要性は、IFU106及びIEU107の両者ともVMU115を共通の、共有の資源として使用することから生じる。アーキテクチャ100の好適な実施例に於いて、仮想アドレスの物理ページ内のアドレスを定義する下位ビットはIFU106によって、制御線102を介してCCU110に直接転送される。IFU106によって供給される仮想アドレスの仮想化上位ビットはバス103、104のアドレス部分によってVMU115に供給され、そこで対応する物理ページ・アドレスに変換される。IFU106にとっては、この物理アドレスは変換要求がVMU115に出された1/2内部プロセサ・サイクル後、制御線111を介してVMU115からCCU110に直接転送される。

【0023】一方、IFU106によって取り出された命令ストリームは命令ストリーム・バス108を介してIEU107に供給される。制御信号は制御線109を介してIFU106とIEU107の間でやり取りされる。IEU107はデータを双方向データ・バス112を介してD\_\_キャッシュ215に格納し、またそれから検索する。IEU107によるデータ・アクセスの場合、物理アドレス全体が制御バス113のアドレス部分によってCCU110に供給される。IEU107はVMU115を、仮想データ・アドレスを、CCU115への送り出しに適切な物理データ・アドレスに変換する資源として利用する。IFU106に対するオペレーションとは異なり、VMU115は対応する物理アドレスをバス104を介してIEU107に返す。

【0024】CCU110はホスト・プロセサ105と主メモリ160との間のバッファとして使用される。一般的に、CCU110は小型の、高速メモリで、ホスト・プロセサ105の近傍に位置し、最も最近アクセスされたコード又はデータを保持する。CCU110は、適切であれば物理アドレスで定義されたデータ要求が命令及びデータ・キャッシュ118、119から満たされるか否かを決定するという一般的に従来の高レベルの機能を行なう。命令キャッシュあるいはデータ・キャッシュ118、119へのアクセスによってアクセス要求が満たせる場合、CCU110はデータ・バス101、113を通じてデータ転送を調整し実行する。命令キャッシュ或いはデータ・キャッシュ118、119へのアクセスによってアクセス要求が満たせない場合、CCU110は対応する物理アドレスをMCU120に供給する。この場合、物理アドレスの他に、主メモリ160への読み出し又は書き込みアクセスが必要であるかを識別するために十分な制御情報、各要求のソース又は行き先キャッシュ118、119、更に要求されたオペレーションがIFU106又はIEU107によって発行された最終的なデータ要求と関連付けられるための追加の識別情報がCCU110によってMCU120



(7)

11

に供給される。

【0025】図2にIEU107のデータ・バスの代表的な高レベルのブロック図を示す。IEU107の目的は最小限の時間で最大数の命令を実行することである。IEU107にはレジスタ・ファイル250、ロード/ストア・ユニット(LSU)205、命令バス(IBUS)225、一式の機能ユニット260、262、230、イミディエート変位バッファ255、セグメント・ベース・ジェネレータ257、及び書き込みバス270が含まれている。LSU205はLSUアドレス・バス220及びLSUデータ・バス210の二つの部分に分かれている。

【0026】スーパースカラ制御ブロック(図示せず)はデータ依存性検査を行い、必要な機能ユニット260、262、及び230が使用可能であるかどうかを検査することによって、ある命令が発行可能であるか否かを決定する。一旦スーパースカラ制御ブロックが一つの命令を発行するように決定すると、IBUS225は発行される命令が必要とするデータを検索する(検索はレジスタ・ファイル250)パイパス・データ280、282、或いはイミディエート・データ258、259から行われる)。IBUS225は複数のマルチプレクサによって構成され、これらのマルチプレクサが、どのデータが機能ユニット260、262、230に転送されるかを選択する。IBUS225はAバスとBバスと呼ばれる一対のバスにデータの転送を行なう。選択されたデータは、諸機能ユニット260、262、230のうちどの機能ユニットがその命令によって使用されるか、或いは現在実行中の命令のオペレーションによって必要とされているか、を決定することによって、AバスかBバスのどちらかのバスに入れられる。

【0027】ほとんどの命令の入力及び出力は複数のレジスタ・ファイルのうち一つのレジスタ・ファイルから送られる、つまり格納されている。好適な実施例では、各レジスタ・ファイル250(例えば、別個の整数、浮動小数点、或いはブール・レジスタ・ファイル)は32個の実数エントリ254及び8個の一時バッファ252のグループを有する。一つの命令が完了すると(「完了」とはオペレーションが終了し、オペランドはその行き先レジスタに書き込める状態にあることをいう)、その結果は一時バッファ252中の事前に割り当てられた場所に格納される。これらの結果は後に実レジスタ254中の適切な場所に移される。このような結果の一時バッファ252から実レジスタ254への移動は「退避」(retirement)と呼ばれる。一度に複数の命令が退避できる。退避により、コンピュータのプログラムカウンタを含めて、マシンの「公式な状態」の更新が行なわれる。

【0028】命令は「バケット」と呼ばれる4個のグループ毎に命令デコードFIFO(first-in-first-out)先入れ先出し方式)レジスタ・スタック記憶装置(図示せず)(本明細書では命令ウィンドウと呼ぶ)を介してIFU106からIEU107へ送られる。バケットはロード、スト

12

ア、及び2個の実行ユニットで構成される4個のユニットに分解される。バケットがこれら4個のユニットに分解された理由はシステム100はロード、ストア、実行の各オペレーション又はそれら全ての組み合わせを実行できる命令を使用して動作するからである。従って、本発明はこれら三つの場合の全てを処理できるバケットを供給するものである。

【0029】IEU107は一度に4個のバケットの命令までデコードしスケジュールできる。命令ウィンドウは全部で16個の命令を4個のバケットに格納する。IEU107は命令ウィンドウを検査し、各サイクルごとにIEU107は命令ウィンドウから最大数の命令を発行しようとする。一旦1個のバケット中の全ての命令が実行され、それらの結果がプロセサのレジスタ・ファイル250に格納されると、そのバケットは命令ウィンドウからフラッシュされ、次に新しいバケットが命令ウィンドウに格納される。一旦その命令が発行されると、レジスタ・ファイル250中の諸レジスタがアクセス可能となる。一時レジスタ252は先行の命令によって生成されたデータに対してデータ依存性を持っていた命令が実行されるとアクセスされる。レジスタ・ファイル250からのデータはデータ線254を介してIBUS225に転送される。

【0030】DAFU230はLSU205によって使用される32ビットのリニア・アドレスを計算する。DAFU230では多数の異なったアドレス指定モードがサポートされている。2サイクルを必要とするデータが4ワード境界を越える場合、そのデータの最初と最後のアドレスはDAFU230によって計算される。アドレスを形成するために4個までのコンポーネントが加算される。すなわち、セグメント・ベース、ベース・レジスタ、スケールド・インデックス・レジスタ、及び変位値、の4個のコンポーネントである。セグメント・ベースには目的のメモリ・セグメントの開始アドレスが含まれている。ベース及びインデックス・レジスタはレジスタ・ファイル250中のどの32ビットレジスタであっても構わない。インデックス・レジスタはそれを1、2、4、又は8で乗算することによってスケールされる。変位値は命令中に存在する定数値(イミディエート値)である。これらのフィールドのうちどのフィールドも省略可能であり、斯くしてアドレス演算に於ける最大限の自由度が得られる。

【0031】セグメント・ベースはセグメント・レジスタ・ブロック257から得られる。セグメント・ベース・ジェネレータ257はデータが如何にしてメモリ中で分割されているかを示す一つの値を生成し、この値をデータ線266を介してDAFU230に転送する。変位はイミディエート変位バッファ255から得られる。イミディエート変位バッファ255はイミディエート・データを線265を介してDAFU230に転送し、またそれぞれデータ線258及び259を介してIBUS225に転送する。DAFU230及びVMU115はLSU205に全てのロード及び/又はストア要求を供給す

(8)

13

る。LSU205はこれらの要求を処理し、後に全ての要求されたデータを書き込みバス270に返す。書き込みバス270はマルチプレクサの集合で成り、マルチプレクサは優先度スキームの基づいてどのデータがレジスタ・ファイル250にラッチするかを選択する（例えば、LSU205によって供給されるデータ又は機能ユニット260 或いは262によって供給されるデータ）。そのデータは線275、276を介して書き込みバス270からレジスタ・ファイル250に転送される。ロード及び/又はストアからのデータは常に最高の優先度与えられる。時折、2個の命令が連続して発行され、それらが相互に依存している場合、IEU107はそのデータをレジスタ・ファイル250に格納することをバイパスし、それを直ちにIBUS225にラッチしようとする。これはデータ線280、281を介して達成できる。従って、データを待つ資源は、そのデータがレジスタ・ファイル250の中を通過するまで待つサイクルを浪費しなくてすむ。

【0032】データ線275、276からのデータも又、一つの命令が実行オペレーション及びストア・オペレーションを伴う場合、LSUデータ・バス210に直接供給される。ロード及び実行オペレーションが行なわれた後、ストアを行うためにデータはLSUデータ・バス210に直接供給できる。斯くしてストア・データを得るために一時レジスタ・ファイル252をアクセスする手間が省け、従って命令の実行時間の増大につながる。LSU205の主な目的は可能な限りCCU110にロード要求をアウト・オブ・オーダーで行い、ロード・データをなるべく速くIEU107に返すことである。ロード・オペレーションはアドレス衝突がなく、書き込み実行待ちが存在しない時のみ実行できる。アドレス衝突が発生するのは、古い命令がまだ書き込み中の記憶場所に於いて読み出しが要求された時である。書き込み実行待ちとは、古い命令が格納オペレーションを要求したがストア・アドレス計算がまだ行なわれていないことである。LSU205はデータ・バス210とアドレス・バス220という二つの部分に分割されている。アドレス・バス220はDAFU230、VMU232、及びCCU110とインタフェースし、データ・バス210は書き込みバス270、CCU110、DAFU230、及びIBUS225とインタフェースする。LSUの三つの主要なタスクは(1) アウト・オブ・オーダーのキャッシュ要求の処理、(2) アドレス衝突の検出、及び(3) データの位置合わせである。

【0033】各命令バケットは同一のアドレスに対するロード及びストア（その間に他のオペレーションが含まれることもある）、ロードのみ、ストアのみ含むことができる。或いはロードもストアも含まないこともある。従って、LSU205は最大4個のロードと最大4個のストアから選択することができる。本発明の好適な実施例で使用する命令セットはCISC型命令セットで、それによって次のような複雑なオペレーションが可能となる。

【0034】a)  $R1 \leftarrow R1 + [R2 + (R3 * 2) + 3]$

14

b)  $[R2] \leftarrow [R2] \text{ OR } R4$ 

但し、 $[x]$  はアドレスxに存在するメモリ・オペランドである。好適な実施例に於ける命令デコード・ユニット（図示せず）はこれらのCISC型命令を次のようにRISC型シーケンスに分解する。

【0035】

a)  $\text{LOAD } [R2 + (R3 * 2) + 3] \rightarrow \text{Temp Register}$ Execute  $R1 + \text{Temp Register} \rightarrow R1$ b)  $\text{LOAD } [R2] \rightarrow \text{Temp Register}$ 10 Execute  $\text{Temp Register OR } R4 \rightarrow \text{Temp Register}$ STORE Temp Register to address  $[R2]$ 

このどちらの場合でも、DAFU230はメモリ・オペランドのアドレスを計算するが、ロード及びストアは同じアドレスを共有しているので1個の命令バケット当たり一つのアドレス計算だけが必要である。CISC型命令をRISC型命令にデコーディングすることについての説明に関しては1992年3月31日出願の米国特許出願番号07/857,599

（代理人整理番号SP032）「CISC型からRISC型命令への変換のためのアライメント並びにデコーディング」（CISC to RISC Instruction Translation Alignment and Decoding）を参照されたい。当該出願の開示を参照することによって本出願に含まれているものとする。

【0036】図3にLSU205のアドレス・バス220の詳細なブロック図を示す。ロード命令は命令ウィンドウから発行され、IEU107によってアウト・オブ・オーダーで実行されるが、ストアは常にイン・オーダーで発行される。ロード及び/又はストア用のアドレスは、全てのオペランドが有効で且つDAFU230がアドレス計算に使用可能になりしだい計算される。DAFU230から物理アドレスを受け取る前にLSU205はキャッシュ要求を行うことができるが、次のクロックサイクルまでに物理アドレスがDAFU230からもVMU115からも来ない場合、キャッシュ要求は取り消される。その場合、キャッシュ要求は後に再発行されねばならない。

【0037】各命令バケット当たり1個のアドレスのみが必要で、そのアドレスはロード・アドレスとストア・アドレスの両方として機能する。各命令バケット当たり、2個の32ビット・アドレスはアドレス・バッファ310～313のうち一つに格納される。すなわち、アクセスの最初のバイトが一つのバッファに格納され、アクセスの最後のバイトが別のバッファに格納される。下位12ビットがDAFU130で準備されると、これらのビットは一時バッファ305にラッチされる。上位20ビットがVMU115で準備されると、次のサイクルで全ての32ビットは適切なアドレス・バッファにラッチされる（すなわち、Address 1又はAddress 2）。アドレス計算は命令の順序で行なわれず、レジスタ依存性が解消した時行なわれる。アドレス変換の後、有効ビット（図示せず）が命令のアドレス・バッファ310～313に設定され、アドレスが有効であることを示す。両方のアドレスは二つの理由で保持

15

される。すなわち、アドレス衝突の検出とページ交差用のキャッシュ要求である。

【0038】IFU106によって使用されるアドレスが仮想アドレスであるのに対し、LSU205によって使用されるアドレスは物理アドレスである。IFU106は、CCU110とVMU115間の調整によって物理アドレスが生成されるのに依りつつ、仮想アドレスに対して動作するのに対し、IEU107ではLSU205が物理アドレス・モードで直接動作することが必要である。この条件が必要である理由はオーバーラップする物理アドレスのデータ・ロード及びストア・オペレーションを伴う、アウト・オブ・オーダーで実行される命令が存在する場合、データの保全を保証するためである。データ保全を保証するために、データがストア命令によって供給された場合、LSU205はストア命令がIEU107によって退避されるまでそのデータをバッファリングする。従って、LSU205によってバッファリングされたストア・データはLSU205にのみ一意的に存在することがある。同一の物理アドレスを実行済みではあるが未だ退避されていないストア命令として参照する複数のロード命令は、ストア命令が実際に退避されるまで遅らされる。その時点で、ストア・データはLSU205によってCCU110に転送可能となり、次に、CCU110のデータ・ロード・オペレーションの実行によって直ちに再びロードされる。

【0039】上述の如く、DAFU230によるアドレス計算は1クロック・サイクルで起こり、VMU132によるアドレス変換は次のクロック・サイクルで起こる。アドレスがロード用のアドレスであるならば、キャッシュ要求が行われる。一方、アドレスがストア用のアドレスであるならば、格納を行う前にLSU205は退避信号が送られて来るのを待つ。ロード要求はCCU110に対してアドレス計算の最初のサイクルでも行なえる。この時点で該アドレスの下位12ビットがCCU110に送られ、上位20ビット（ページ番号を表わす）はアドレス変換の後、次のサイクルでCCU110に送られる。ロード・ストア・アドレス・バス220が使用可能な場合、線330を介してイミディエイト要求をキャッシュ110に対して行なうことができる。現在、ロード・ストア・アドレス・バス220には実行待ちのロード及び/又はストア・アドレスは存在しないので、アドレス衝突の可能性も書き込み実行待ちの可能性も全く存在しない。従って、直ちにキャッシュ110に対して要求を行なえる。

【0040】ブロック340には複数のマルチプレクサが含まれているが、このブロックはアドレス・バッファ310～313からキャッシュ要求用のアドレスを選択するために使用される。LSU205はキャッシュ110に対して要求を行なうためにアドレス・バッファ310～313（即ち、予約ステーション）を使用する。4個のアドレス・バッファ310～313（予約ステーションとも呼ばれる）は中央命令ウィンドウ（図示せず）に含まれる4個のバケットに対応する。IEU107がデコード・ユニット（図示せず）

(9)

16

から新しいバケットを要求すると、アドレス・バッファ310～313のうち一つが予約される。アドレス・バッファ310～313は命令番号に従って割り当てられる。最も若い（最新の）命令を示すための履歴ポインタが更新される。この時点に於いて、命令がロード、ストア、その両方を伴うものであるか、あるいはそのどちらも伴わないものであるかが判明する。また、ロード及び/又はストアオペレーションで使用されるデータのサイズも判明する。対応する命令がIEU107によって退避された時に、アドレス・バッファ310～313は割り当て解除される。割り当て解除の後、新しい命令バケットがデコードユニット（図示せず）から受け取られる。ロード・バイパス及びアウト・オブ・オーダー・ロード実行を使用するためには、ストアに対するロード依存性（アドレス衝突）を検出する必要がある。ロード依存性はアドレス衝突または実行待ちのストア・アドレスのよって示される。ロード依存性が発生するのは、古い命令がストア・オペレーションを要求した記憶場所と同じ記憶場所でロード・オペレーションが要求された時である。アドレス衝突の検出には、ロードの最初のアドレスが各先行ストアの2個のアドレス（最初と最後）と比較される必要がある。アドレスの最後のバイトとのこのような比較が必要なのは、ストアが4ワード・ページ境界を越えたり、位置合わせがなされていなかったりするからである。アドレス・ビットのマスキングは偽の依存性検出を最低限に抑えるためにデータのサイズに応じて行なわれる。ロード・データが4ワード（64ビット）境界からはみだすと、好適な実施例ではそのロード・データにはロード依存性があると仮定される。その理由は、ロードの第2アドレスを各ストアの二つのアドレスと比較するコンパレータが存在しないからである。一つの衝突が検出されると、ロード・オペレーションはその衝突しているストア・オペレーションがCCU110に送られるまで待たなければならない。実行待ちのストア・アドレスとはストアのアドレスが未だ有効でないということを意味する。従って、そのアドレスが判明するまでロード依存性の存在が仮定されるのである。

【0041】図4にLSU205によって使用されるアドレス衝突ブロック400の概略図を示す。アドレス比較論理による二つのアドレスの比較は最下位ビットのビット0～4がマスクされた後行なわれる。マスキングの後、アドレスが全く一致するならば、これら二つのアドレスの間に衝突が存在することになる。各比較ごとに、二つのオペレーションの内最大のオペランドがマスキング制御のために使用される。各アドレスから、0～4個の最下位ビットがマスクされる。その際、回路400はアドレス・バッファ410～413の各バッファごとに1回、つまり合計4回複製される（図4にアドレス・バッファ310のアドレス衝突検出ブロックを示す）。

【0042】各ロードの最初のアドレス405、406がア

(10)

17

ドレス407～418のうち1対おきに比較される。2個の比較の結果とその有効ビット419～424間でANDがとられ、その後全部でORがとられ、その結果アドレス・マッチ430a, 430b, 430cが生成される。アドレス・マッチ430は次に命令番号比較425～427及びストア・ビット431～433とANDがとられ、その結果衝突チェック450a, 450b, 450cが生成される。命令番号比較425～427は二つの命令間の比較的新しさを示す。例えば、命令番号比較425はアドレス・バッファ310中の最初の命令と、アドレス・バッファ311中の最後の命令との間の比較的新しさを示す。第2命令が第1命令よりも古ければ、衝突は存在しない。これら3個の衝突検査はORがとられて、検査を受けている特定のロードのアドレス衝突信号460を生成する。

【0043】アドレス衝突の検出に於いては、各ロードの開始(第1)アドレスが各ストアの第1及び第2アド\*

オペランドサイズ

マスクするビット数

1バイト

0ビットマスク

2バイト

アドレスが0で終わる場合、1ビットマスク

アドレスが01で終わる場合、2ビットマスク

アドレスが11で終わる場合、3ビットマスク

4バイト

アドレスが00で終わる場合、2ビットマスク

アドレスが1又は10で終わる場合、3ビットマスク

8バイト

3ビットマスク

10バイト

4ビットマスク

更に、ロード・オペレーションが4ワード境界を越えるたびに、アドレス衝突が発生していることが假定される。その理由は、ロードの最初のアドレスだけがストア・アドレスと比較されるので、アドレス衝突が検出されないことがあるからである。ハードウェア中で使用されるコンパレータの数を二倍に増やすことにより、この制\*

例1:

オペレーション	アドレス1	アドレス2	サイズ	マスク
LOAD	…1001	…	2バイト	2ビット
STORE	…1000	…1011	4バイト	2ビット

ロード・アドレス1001がマスクなしに1000及び1011と比較された場合、ストアがバイト1000, 1001, 1010, 及び1011に書き込んだとしても衝突は検出されない。2個のLSBがマスクされていれば、結果は次のようになる。★

例2:

オペレーション	アドレス1	アドレス2	サイズ	マスク
LOAD	…0100	…	4バイト	2ビット
STORE	…0001	…1000	8バイト	3ビット

3個のMSBがマスクされていれば、下記のアドレスが生成され、アドレス衝突が検出される。

【0047】

オペレーション	アドレス1	アドレス2
LOAD	…0000	…
STORE	…0000	…1000

2個のLSBだけがマスクされているならば、下記のアド

18

\*レスと比較される。一つのロード又はストア・オペレーションは1～10バイトまでのどこかをアクセスするので、衝突が検出されることを保証するためにそれらのアドレスのうち幾つかがマスクされる。このようなマスクングは信号470～475で行なわれる。二つのアドレスが相互に比較される前に、最下位ビットのうちビット0, 2, 3, 又は4がマスクされる。マスクされたアドレスが全く一致する場合(等しい比較)、アドレス衝突の可能性はある。マスクされるビットの数(0, 2, 3, 4)はアドレスが比較されているその二つのアドレスのオペランドのサイズ、そして第1アドレスの最下位の2ビットによって異なる。第1アドレスの最下位2ビットが使用されるのは、間違っ検出される衝突の数を制限するためである。マスクングに於いて、最大のオペランド・サイズは次のように使用される。

【0044】

※約は削除できる。ストア・アドレスが4ワード境界を越えることがあれば、アドレス衝突は検出される。

【0045】マスクングの必要性を次の幾つかの例で示す。(下記の全ての数字は二進数である)。ロードのアドレス2は衝突検査の目的で使用されないの、アドレス2は省略する。

★【0046】

オペレーション	アドレス1	アドレス2
LOAD	…1000	…
STORE	…1000	…1000

40

レスが生成され、アドレス衝突は検出されない。

【0048】

オペレーション	アドレス1	アドレス2
LOAD	…0100	…
STORE	…0000	…1000

前述の如く、LSU205はキャッシュ要求を必要とする最大4個のロード命令と最大4個のストア命令のウィンドウ

50

(11)

19

から選択を行なうことができる。これらのロード及びストアはCCU110に対して互いに競合し、競合するロード及びストア間の選択は下記の如く行なわれる。

【0049】ストア命令は、単に他のロード及びストアだけではなく、全ての命令に対してプログラム順序で行なわれなければならない。ストア要求はストア命令を退避する信号がIEU107から送られた段階でCCU110に発行される。この信号は、全ての先行命令が終了し、それらの命令では例外も、誤って予測された分岐も無かったことを知らせる。ストア命令をこの信号よりも早く行なうことは不可能である。その理由は、ストアはマシンの状態を非可逆的に変更するので、例外も分岐も発生しなかったことを確認することが重要であるからである。データ・キャッシュ119の使用の目的ではストアはロードに優先する。その理由は、ストアの遅延はバケットの退避の遅延をもたらし、従って命令デコード・ユニット（図示せず）からの次のデコード済みバケットの受理を遅らせるからである。

【0050】ロードが先行ストアに依存しない限り、ほとんどのロード命令はアウト・オブ・オーダーで発行できる。この例外は、メモリ・マップされたI/Oからの読み出しのような、副作用を持つロードである。本発明の好適な実施例ではメモリ・マップ入出力（I/O）サブシステムが使用される。ある種のI/Oデバイスは読み出しによってアクセスされるとその状態が変化する。例えば、ある種のFIFOバッファは次のデータ項目に順番を付けて、その結果ある種のデバイス状態レジスタは自動的にクリアされる。このようなシステムに於いては、ロード・バイパスは危険なオペレーションである。誤って予測された分岐、或いは例外のために、バイパスされたロードが誤って発行されることがある。そのようにバイパスされたロードがシステム状態を不正に変更するような事態が生じてはならない。

【0051】この問題を解決する方法はこれらの要求がイン・オーダーで行なわれるようにロード/ストア・ユニットを構成することである。ロード/ストア・ユニットでは、キャッシュ要求で要求されたデータがキャッシュ可能であるか否かにかかわらずキャッシュ110に通知を行なう機構が用意されている。この機構によって、プロセッサはこのデータがライト・スルーである、つまりキャッシュ可能である、とキャッシュ110に通知することができ、また直ちにメモリはライト・スルーを行なうべきであると通知する。システムの状態を変更する外部読みだしアクセスはこれらのキャッシュ不能アクセスのサブセットであるが、上記の問題は、このデータはキャッシュ不可能であるとのキャッシュ110への通知に関連してイン・オーダーの要求を行なうことによって解決される。従って、ロードバイパスを完全に回避するかわりに、プロセッサはキャッシュ不可能なロードのバイパスを防止できる。このようにして、ほとんどのロード・オペレーシ

20

ョンが、まれに発生するキャッシュ不可能なロードでの不正なオペレーションを生成することなく、バイパスを利用できるようになる。このような機構はまたメモリ変更以前に例外が発生しないことを保証するためにも必要である。一つのロードが一つのストアに対して依存性を持たない場合、「ストアのロード・バイパス」が発生する。各ロードはページ・キャッシュ使用不可（page-cache-disable）及びページ・ライト・スルー（page-write-through）という2個のビットと対応している。これらのビットはVMU115又はIEU107から得られるビットである。

【0052】ストア・データは二ヶ所のうちの一つから生成される。第1に、それは64ビット整数ストア中に整数データ・バス上でLSU205に直接発行できる。第2の方法は整数及び浮動小数点機能ユニットによる結果を監視（スヌーピング）することによって行なわれる。これは通常の「実行後格納」シーケンスをサポートするために行なわれる。このシーケンスでは一つの命令の実行の結果はその命令のストア・データである。そうすることによって、“[R2]<[R2]OR R4”のようなCISC型命令の結果が、その命令が明示的にLSU205に発行されなくても格納されるようになる。

【0053】LSU205はサイクルごとに一つの要求だけをCCU110に対して行なうことができ、その場合ストア・オペレーションが優先される。書き込み制御がLSU205に対し、この命令は退避可能であると通知すると直ちにストア・オペレーションはCCU110に送信される。次の優先度はアドレス・バッファ310～313に有効なアドレスを持ち、アドレス衝突も実行待ちの書き込みも持たない、最も古いロード・オペレーションに与えられる。命令間の比較的な新しさはバッファの位置とバッファ・ポインタの値で決定される。最後に、DAFU230から送信された新しいロードが優先度を持つ。この最後の場合、アドレス衝突及び実行待ち書き込みは要求が行なわれるまで検査されず、そして必要ならばロード要求は取り消される。

【0054】時折、キャッシュ・ミスが起こる。ストアの場合、CCU110はこのような事態を処理し、その結果LSU205はキャッシュ・ミスの影響を全然受けずに済む。ロードの場合、LSU205はキャッシュ・ミスについて通知を受け、データが返される前に遅延が起こる。LSU205は次にキャッシュ・ミスの発生をIEU107に通知し、その結果このデータを待っている命令は取り消される。目的のデータがキャッシュ・ライン境界を越えると、ロード・オペレーションに対して2個又は3個のキャッシュ・アクセスが必要になります。これらの要求は連続して行なわれ、1サイクル当たり一つの要求が行なわれる。好適な実施例に於いて、一つのキャッシュ・ラインの幅は8バイトで、000で終了するアドレスに位置合わせされている。3個のキャッシュ要求が必要とされるのは111で終了するアドレスで始まる80ビット・データの場合だけで

(12)

21

ある。このデータがデータ・キャッシュ119 から返される場合、ロード・アライナ550 (図5、図6を参照して下記に説明する) が配置され、このデータのシフトとラッチが行なわれる。

【0055】ほとんどのロード/ストア・ユニットはデータが行き先レジスタに入るようにそのデータをゼロまたはサインで拡張するが、本発明の好適な実施例では、行き先レジスタの初期値が保持され、その一部のみが変更される。勿論、これは8又は16ビット長の整数ロード・データの場合のみ意味がある。レジスタの初期の内容はアドレス計算の時点でLSU 205 に送られ、次にデータ・キャッシュ119 からのロード・データは初期値データとマージされる。

【0056】図5、図6にLSU 整数データ・バス210 の概略図を示す。LSU データ・バス210 はロード及び/又はストア・データをCCU110及びIEU107間で転送する。ロード・オペレーション中に、データは線290 を介してデータ・キャッシュ119 からLSU データ・バス210 に入り、ストア・オペレーション中には線275、276、277 を介してIEU107から入る。データ線275 及び276 は32ビット・データを書き込みバス270 を介して機能ユニット260 及び262 からLSU データ・バス210 に供給し、線282 は有効アドレス又ははマージされたデータを供給する。有効データがLSU データ・バス210 へ供給されるのは一つの命令の結果が、そのアドレス・ロケーションに存在するデータではなく、アドレスそのものである場合である。ストア・データ線516 は64ビット・データをLSU データ・バス210 に供給する。データはデータ線290 または292 を介してデータ・キャッシュ119 又はIEU107のいずれかにそれぞれ返される。

【0057】データ・バッファ520 ~526 は、データ・キャッシュ119 への或いはデータ・キャッシュ119 からのデータ転送中、ロード及び/又はストア・データを保持するために配置されている。各データ・バッファ520 ~526 及びアドレス・バッファ310 ~313 の間に1対1の対応が存在する(そしてこれらのアドレス・バッファ及び4個の命令バケットとの間にも1対1の対応が存在する)。各アドレス・バッファ310 ~313 にはLSU データ・バス210 中に2個の対応するデータ・バッファが存在する。すなわち、整数ロード及び整数ストア・データ(8バイト) 520 ~526 に対して一つのデータ・バッファと、浮動小数点ロード及びストアのデータ(10バイト) 540 ~546 に対して一つのデータ・バッファである。本発明に於いては、浮動小数点演算用の一つの別個のLSU データ・バスが存在する。浮動小数点データ・バッファ540 ~546 の動作は整数データ・バスに関して説明された動作と同一である。一つの命令は整数命令或いは浮動小数点命令のいずれかであるので、この二つのユニットは物理的に接続されていなくても構わない。以下に、整数データ・バッファ520 ~526 の動作のみを詳し

22

く説明する。

【0058】制御線581 及び587 はデータ・フローをそれぞれマルチプレクサ560 及び565 を介して制御するために配置されている。又、制御線582 及び586 はデータ・バッファ520、522、524、及び526 へのデータ・フロー、そしてデータ・バッファ520、522、524、及び526 からのデータ・フローを制御するために配置されている。ロード・オペレーションに於いては、データは線290 を介してデータ・キャッシュ119 からLSU データ・バス210 に入る。ロード・データはアライン・ブロック550 に入り、アライン・ブロックはデータの位置合わせを行ない(下記の説明を参照されたい)、位置合わせされたロード・データをマルチプレクサ530 ~536 に転送する。位置合わせされたロード・データは次に、どの命令がデータを要求したかにより、データ・バッファ520 ~526 の一つにラッチされる。ストア・オペレーション中、ストア・データはデータ線275、276、277 を介してIEU107からLSU データ・バス210 へ入り、その後、データ・バッファ520 ~526 のうち適切なデータ・バッファにラッチされる。

【0059】ロード及び/又はストア・データのうちのどちらかがデータ・バッファ520 ~526 にラッチされると、そのデータは線290 を介してデータ・キャッシュ119 へ、或いは線292 を介してIEU へのいずれかに送られる。4個のデータ・バッファ520 ~526 はデータをマルチプレクサ560、565 に供給し、次にこれらのマルチプレクサはLSU データ・バス210 から転送されるべき適切なデータを選択する。しばしば、ストアを含む命令の結果は主メモリ260 に格納されなければならない。従って、命令の実行の後、その結果はデータ線275、276 を介してLSU データ・バス210 に直接書き込まれる(最初に結果をレジスタ・ファイル250 に格納するのではなしに)。LSU データ・バス210 は命令の退避信号を受け取るまでデータを適切なデータ・バッファ520 ~526 に保持する。

【0060】定期的に、一つの特定な命令は一つの行き先レジスタ全体に格納を行なわなくなっている。この場合、「マージ・データ」がデータ線282 を介してLSUデータ・バス210 に供給される。例えば、一つの命令が8ビットだけを行き先レジスタに格納したいが、残りの24ビットをレジスタに保存したい場合、マージ・オペレーションが行なわれる。従って、データ線282 は行き先レジスタの初期値(すなわち、マージ・データ)をLSU データ・バス210 に供給する。マージ・データ(すなわち、行き先レジスタの内容)は適切なデータ・バッファ520~526 にラッチされる。次に、新しい(ロード)データが線290 (a) を介してキャッシュから戻され、アライン・ブロック550 に入る。アライン・ブロック550 はデータの位置合わせを行ない、それをマルチプレクサ530 ~536 に供給する。ロード・データは次に、マージ・

(13)

23

データを保持している同じデータ・バッファ520～526にラッチされる。一旦全てのデータがアセンブルされると、それは適宜な記憶場所（すなわち、データ・キャッシュ119又はレジスタ・ファイル250）に転送可能となる。

【0061】従来のロード・ストア・ユニットでは普通、アドレスが特定の境界に位置合わせされなければならない。例えば、32ビット・データ・アクセスでは000で終わるアドレスがなければならない。しかしながら、好適な実施例のコンピュータ・アーキテクチャによって8、16、32、64、又は80ビット・データの位置合わせされていないアクセスが可能になる。位置合わせされていないアドレスを有することは次のような影響を及ぼす。つまり、(1) ストアに対するロード依存性検出のためにさらに別のハードウェアが必要である。(2) データがページ境界を越えるとアドレス変換が2回必要になる。(3) 1回のロードに対して複数のキャッシュ・アクセスが必要になる。

【0062】CCU110によって返されたロード・データの長さは8バイトであり、それはデータ・バッファ520～526中の適切な位置に位置合せして格納される必要がある。時には、完全なロードができあがるまでに2又は3のデータ集合が返されねばならない（例えば、二つ以上のキャッシュ・アクセスが必要な時）。更に、これらのデータ集合がアウト・オブ・オーダーで返される場合があるので、特別な措置が必要である。整数データの位置合わせは8個の8入力マルチプレクサ（8ビット幅）を使用して処理される。各マルチプレクサはデータ要求の1バイトに対応する。CCU110からロードされた8バイトのデータのうちのデータが適切なデータ・バッファ520～526にラッチされるべきかを決定するために8ビットの選択線が使用される。更に、データ・バッファ520～526はどのバイトが上書きされるべきかを制御するためにバイト・イネーブルになる。

【0063】図7にキャッシュ線交差を持つ位置合わせされていない整数ロードの1例を示す。この例では、アドレスXXXXXXXXX5から4バイトのロードが要求されているが、このロード要求はキャッシュ線からはみだすので、その結果2個のロード要求が必要とされる。最初のキャッシュ要求がデータを返した後、データはロード・アライナ550に転送される。ロード・アライナ550は最後の3バイトをバイト0までシフトし、その後、最後の3バイトは適切なデータ・バッファ520～526にラッチされる。データ・バッファの最後のバイトはストアによって上書きされない。一旦第2のキャッシュ要求のデータが返されると、図示されているようにキャッシュ線の最初のバイトがデータ・バッファの最後のバイトにラッチされる。更に、この例ではキャッシュ線はイン・オーダーで返されるが、それはどの順序で返されても構わない。

24

【0064】浮動小数点データ位置合わせは整数位置合わせと同じ働きをするが、浮動小数点データ位置合わせの場合、10個の8入力マルチプレクサが使用される。LSU205ではロード・フォワーディングはサポートされていない。ロードがストアに依存する場合、そのロードはロード要求を行なう前に、ストア・データがキャッシュに書き込まれるまで待たなければならない。しかし、本発明の設計では、本質的にロード・フォワーディング機構の実現を阻止するような制約はない。当業者にとって、ロード・フォワーディングを実現するために必要なハードウェア変更を行なうことは容易であろう。

【0065】LSU205の好適な実施例では多重処理環境がサポートされている。各命令はロード及び/又はストア以外に、ロック或いはアンロック・コマンドを含むことができる。これらの信号はキャッシュに送られ、キャッシュはデータをロックし、メモリ及び入出力サブシステムに同じことをするように通知を送る。ロック又はアンロック・コマンドが命令ウィンドウに存在する場合、ロードはそれらの命令の順序と同じ順序で行なわれなければならない。すなわち、後続のロードは最初にロック/アンロック・コマンドを伴うロードを先ず行なわないと実行できない。LSU205のオペレーション例表AにLSU205のオペレーションを示すサンプル・プログラムを示す。プログラムはインテル486（Intel 486）の表記法で記述されている。3個のレジスタが使用され、それらはeax、ebx、そしてecxとラベルされている。ロードされ、ロード及び/又はストアされるデータは32ビット幅のデータであると仮定される。ブラケットにアドレス・ロケーションを示す。

【0066】表A

(1) mov ebx, [ecx]

(2) dec ebx

(3) or [eax], ebx

(4) (size\_16) mov ebx, [eax+3]

このコードの最初の行では、アドレスecxに格納されたデータがebxに移される。従って、この命令は一つのロード・オペレーションである。第2の命令ではレジスタebxにある値が減少され、この命令ではロードもストアも行なわれない。第3の命令はアドレスeaxに格納されたデータ及びデータebxに対して論理和をとり、結果を[ebx]に格納する。従って、このオペレーションではロードとストアの両方が行なわれる。最後に第4命令ではアドレスeax+3に格納された16ビットのデータがebxに移動される。従って、この命令ではロード・オペレーションが行なわれる。

【0067】このコードが実行される前に、下記の値（全て16進法で表記）がレジスタ及びメモリに含まれていると仮定する。

【0068】

50

(14)

25

表B

```

e a x = 0000_0100 [0100] = 0000_4321
[0104] = FFFF_FFFF
e c x = 0000_1201 [1200] = 6500_01FF
[1204] = FFFF_FF87

```

表Aの命令の実行の結果を表Cに示す。

\* \* 【0069】

表C

```

mov ebx, [ecx ]      EBX <... [1201] = 8765_0001
dec ebx              EBX <... 8765_0001-1=8765_0000
or [eax ], ebx       EAX <... 0000_4321 or 8765_0000 = 8765_4321
(SIZE_16) mov ebx, [eax+3 ] EBX <... [0100+3] = [0103]
                        = FF87 ... > 8765_FF87

```

次に、表Aの命令の実行の結果の詳細を説明する。

【0070】図8から図15において、LSU205の代表的な例を示す。各図は一つのサイクルを表わす（例えば、図8はサイクル1を表わし、図9はサイクル2を表わす）。4個のアドレス・バッファ310～313、及びロード701、ストア702、及び有効ビット717が図示されている。更に、衝突ビット710、実行待ちビット715、及び要求されたデータのサイズ指定705が図示されている。アドレス0100から0107及びアドレス1200から1207の  
20 カレント・メモリ内容は参照番号780として示されている。ブロック730にカレント・キャッシュ要求を示す。ブロック740はデータが（そのようなデータが存在するならば）CCU110から最近返されたことを示す。ブロック760はVMU115から返されているアドレスを示し、ブロック770はDAFU230から返されているアドレスを示す。ロード及びストア・ビットはイン・オーダで設定されるが、各アドレスはLSU205にアウト・オブ・オーダで供給されても構わない。ブロック750に、返されたデータが  
30 如何にして物理的に位置合わせされるかを示す。

【0071】図8に於いて、最初の命令は「mov ebx, [ecx]」である。最初に、ecxに格納されているデータはLSUアドレス・パス220に転送されなければならない。ecxに格納されているアドレス、つまり1201はDAFU230から一時アドレス・バッファ305に転送されるが、このアドレス全体は必要でない。最初の12ビットと最下位の3ビットが一時バッファ305に転送される。その理由は、上位20ビットはDAFU230からVMU115に転送されるからである。movオペレーションにはロードが伴うので、バケット0に於けるロード・ビットは1に設定される。要求されたデータは32ビットである（ブロック705の011によって示されている）。これはLSU205中のアドレスの第1集合だから、ブロック730に示すように、アドレス情報は識別子(id)と共に直ちにCCU110に送られる。LSU205は識別子に基づいて、返されたデータがどの命令と対応しているかを決定する。LSU205に転送されるためにアドレスがVMU115によって変換されるのをLSU205が待っている間、一時レジスタ305が使用される。第2命令の「dec ebx」はアドレス・バッファ・キューに  
50

26

入れられる。decオペレーションにはロードもストアも伴わないので、アドレス・バッファ311に対応するロード・ビット701とストア・ビット702の両方が0に設定される。ロードもストアも必要でないので、この命令ではアドレス計算は必要でない。

【0072】図9に於いて、ecxに格納されたアドレスの第1バイトはレジスタ・アドレス1(address1)に入れられ、そのアドレスの最後のバイトはアドレス2(address2)に入れられる。これらのアドレスは両方とも勿論アドレス・バッファ310中に存在する。両方のレジスタ(address1及びaddress2)は有効なアドレスを含んでいるので、両方の有効ビットが設定される。address1及びaddress2が異なった時点でアドレス・バッファ310～313にラッチされることも可能である。これが発生するのは、VMU115からの変換を2回必要とするページ・クロッシングが起こった時である。第3の命令は「or [eax], ebx」である。第3命令に関する(IEU107から送られる)情報はアドレス・バッファ312に対応する適切なブロックに入れられている。OR命令はロード及びストア・オペレーションを必要とするので、両方のビットとも適宜に1に設定されている。要求されたデータの長さは32ビットで、ブロック705に示されている。更に、第3命令に対応するロード/ストアのアドレスはブロック770に示すようにDAFU230から供給される。そして、ブロック730に示すように、このデータに対してキャッシュ要求が行なわれる。更に、第2サイクル中に、第1命令用に要求されたデータはキャッシュから検索され、データ・レジスタ520に格納されている。しかし、ブロック730に示される、返されたデータは位置合わせされていないデータである。CCU120はアドレス1200で始まるデータのブロックを返したが、命令が要求したデータは1201で始まる32ビットのデータである。従って、ブロック750に示すように、データは位置合わせされなければならない。返されたデータはLD-ALIGNを00000010に設定することによって2ビット分シフトされ、最初の32ビットのデータはBYTE-SELによって選択される。

【0073】図10に於いて、ブロック770に示すように、次のアドレスがDAFU230によってLSU205に供給され



(15)

27

る。第3命令に対応するアドレスはアドレス・バッファ312にラッチされる。有効ビット717の両方のビットが設定される。第1命令がそのオペレーションを完了したので(すなわち、データがCCU110から返され、IEU107に送られたので)、今や有効ビットがリセットされている。(バケット番号が4にリセットされているのは例示の目的のためである。好適な実施例に於いては、ポインタが命令の比較的新しさを管理するために使用される)。第3命令ではeaxに格納されたアドレスの取り出しが必要である。アドレスが一旦LSU205に入ると、キャッシュ要求が実行可能となる。更に、第4命令に関する情報、すなわち、その命令はロードで要求されているデータの幅は16ビットである(010によって示されている)が、アドレス・バッファ313に対応する適切なブロックで示されているように、この情報がIEU107から送られて来ている。しかし、第4命令より古いストア(すなわち、第3命令)が存在する。LSU205はポインタを使用して、どのアドレス・バッファが最も古い命令情報を含んでいるかを決定する。このストアが存在するので、アドレス・バッファ313に対応する書き込み実行待ちビット715が設定される。従って、この場合、第4命令用のキャッシュ要求は生成できない。ブロック740に示すように、CCU110は第3命令用にデータをLSU205に戻す。要求データはアドレス100で始まっているので、戻されたデータは位置合わせされる必要がない。最初の32ビットだけがBYTE-SELで選択され、そしてデータはデータ・バッファ526にラッチされる。

【0074】図11に於いて、第4命令に対応するアドレスはアドレス・バッファ313にラッチされ、対応する有効ビットが設定されている。次に、アドレス衝突オペレーションが行なわれる。第4命令からのaddress1が第3命令のaddress1及びaddress2と比較され、その結果アドレス衝突の存在が決定される。従って、アドレス・バッファ313に対応する衝突ビット710が設定される。衝突が存在するために、サイクル4の期間中キャッシュ要求は生成できない。しかし、キャッシュ要求の実行が不可能であっても、ブロックで示されているように第4命令用のマージ・データがIEU107から到着する。マージ・データとはレジスタebxからのデータである。マージ・データが必要なのは、第4命令が単に16ビット・オペレーションであるからである。このマージ・データはデータ・バッファ526にラッチされる。

【0075】図12に於いて、書き込みAデータ(WRA-DATA)がIEU107から到着する。WRA-DATAは第3命令に於けるOR演算の結果である。このデータはデータ・バッファ524にラッチされる。更に、ブロック780に示すように、サイクル5の期間中次のバケット、即ちバケット1、が退避される。具体的には、retire next ビットが1に設定され、次の命令が退避可能であることを示し、retire num ビットが1に設定され、バケット1にある

28

命令を退避すべきであることを示す。この場合、第3命令と4命令の間にアドレス衝突がまだ存在する。

【0076】図13に於いて、データ・バッファ524中のデータはebx中のデータとORがとられ、その結果値87654321が生成される。ブロック785に示すように、サイクル6の期間中、第3命令が退避される。第3命令の退避によって、LSU205は第4命令に対応する衝突ビット710をリセットできるようになる。ブロック730に示すように、OR演算によって生成された値を記憶場所00000100(レジスタeaxに格納されているアドレス)に格納するためのキャッシュ要求が行なわれる。ブロック780に示すように、データはこのデータ・ロケーションに格納されている。

【0077】図14に於いて、第4命令は記憶場所0103(レジスタeax+3の最初の16ビット)に格納されているデータをロードする。従って、ブロック730に示すように、第4命令に対応するロード・オペレーション用にキャッシュ要求が行なわれる。

【0078】図15に於いて、ブロック740に示すように、要求された(位置合わせされていない)ロード・データがキャッシュから返される。ブロック750に示すように、次にデータは3バイト分シフトすることによって位置合わせされる。その理由は、要求されたデータはアドレス0100でなくアドレス0103で始まるからである。最初の16ビットだけが要求されたので、最初の2バイトだけが位置合わせされたデータから選択される。これら16ビットは次にデータ・バッファ526にラッチされ、このデータ・バッファはIEU107に逆方向に転送されて、レジスタebxに格納される。上記に本発明を実施例を参照しつつ説明したが、本発明の精神及び特許請求の範囲から逸脱することなく、形状並びに詳細において様々な変更が可能なのが当業者には理解されるであろう。

【図面の簡単な説明】

【図1】 本発明が動作するマイクロプロセッサ・アーキテクチャのブロック図である。

【図2】 ロード・ストア・ユニットを含む命令実行ユニットを示す一般的なブロック図である。

【図3】 LSUアドレス・バス220を示すブロック図である。

【図4】 LSUに位置するアドレス衝突ブロックを示す概略図である。

【図5】 LSUデータ・バスを示す概略図である。

【図6】 LSUデータ・バスを示す概略図である。

【図7】 キャッシュ線交差を持つ位置合わせされていない整数ロードの例を示す図である。

【図8】 LSUの動作の一例を示す図である。

【図9】 LSUの動作の一例を示す図である。

【図10】 LSUの動作の一例を示す図である。

【図11】 LSUの動作の一例を示す図である。

【図12】 LSUの動作の一例を示す図である。

(16)

29

【図13】 L S Uの動作の一例を示す図である。

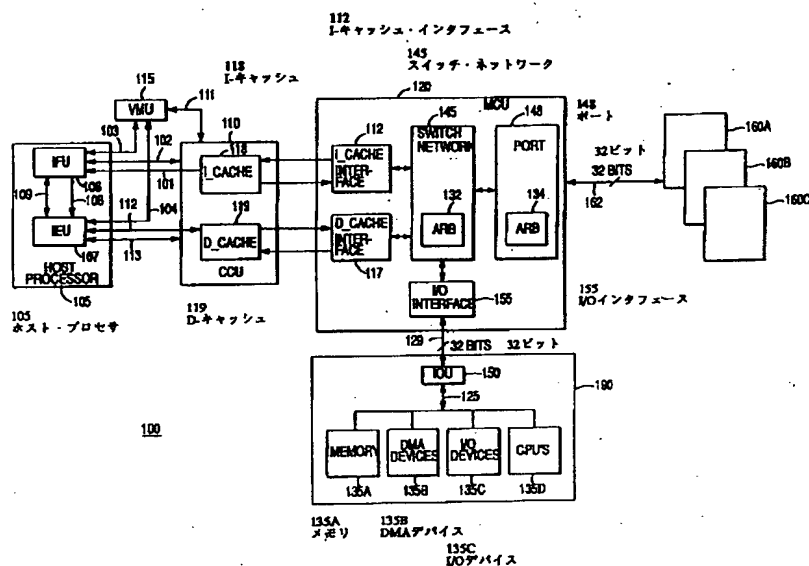
【図14】 L S Uの動作の一例を示す図である。

【図15】LSUの動作の一例を示す図である。

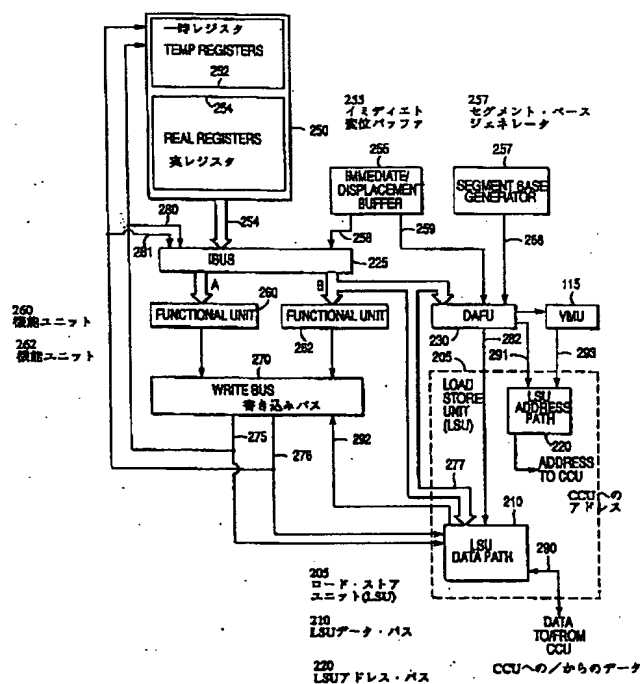
【符号の説明】

100 …マイクロプロセサ・アーキテクチャ、105 …ホス

【図 1】



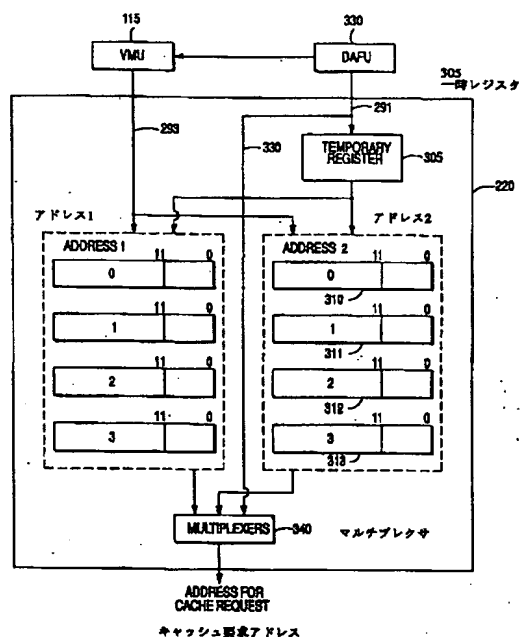
【図 2】



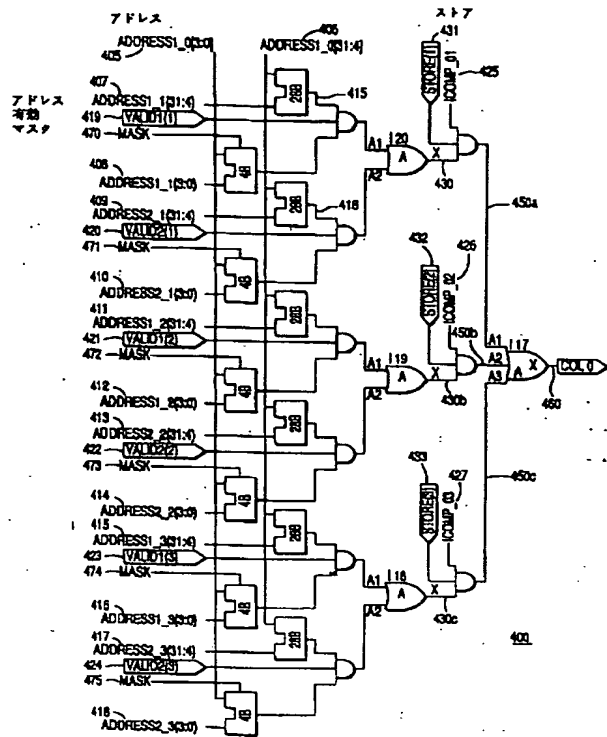
30

ト・プロセサ、110 …キャッシュ制御ユニット及びメモリ (CCU)、115 …仮想メモリ・ユニット、120…メモリ制御及びインタフェース・ユニット、135 …周辺入出力装置、160 …主メモリ、160a、160b、160C…インタリーブド・メモリ・バンク、190 …入出力サブシステム。

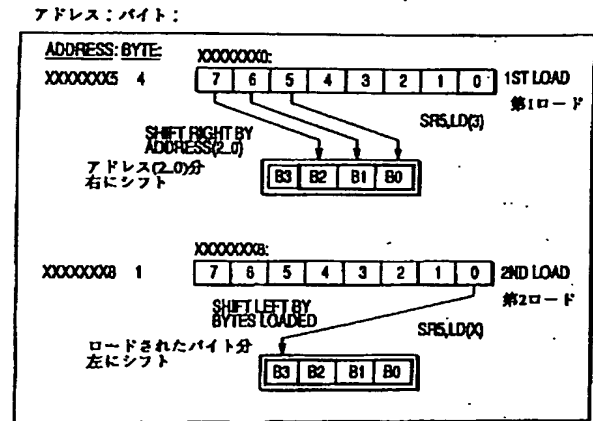
【図 3】



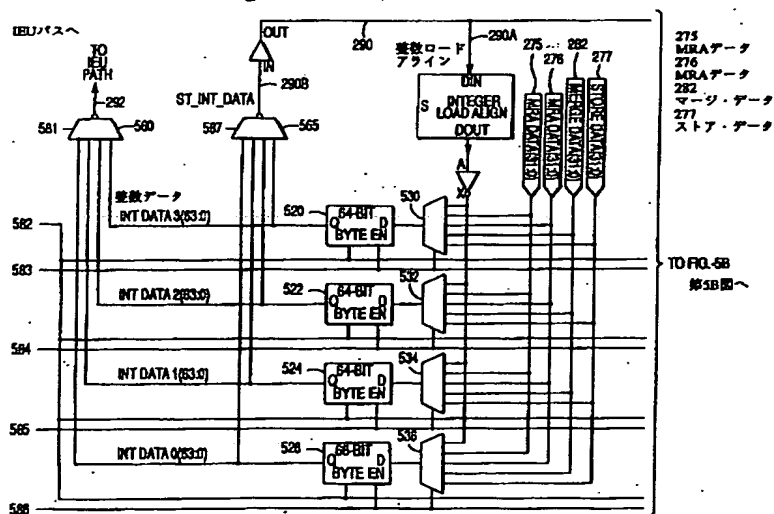
【圖 4】



【図 7】

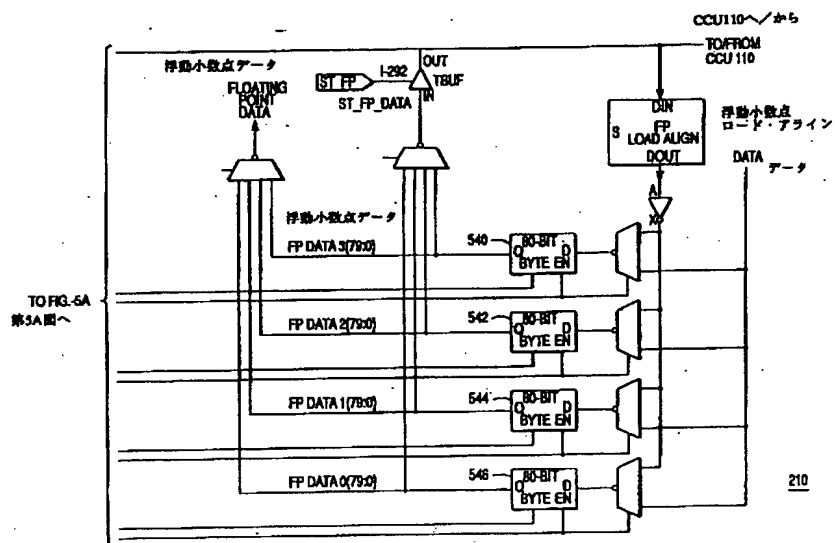


【図 5】

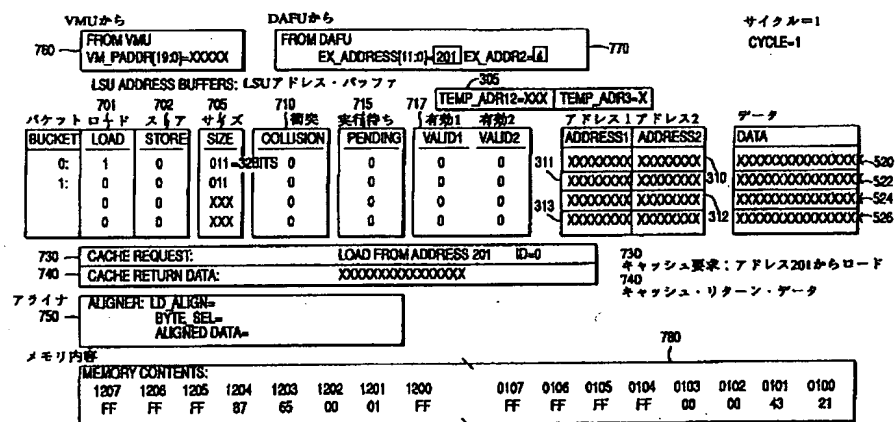


(18)

【図 6】

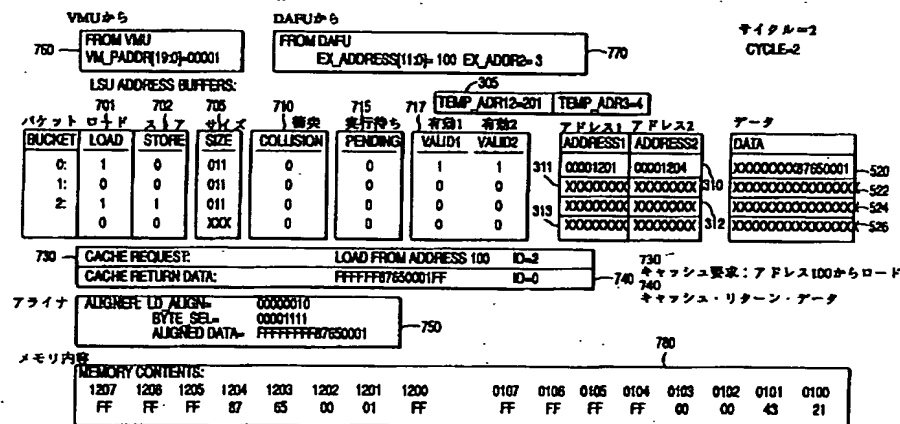


【图 8】

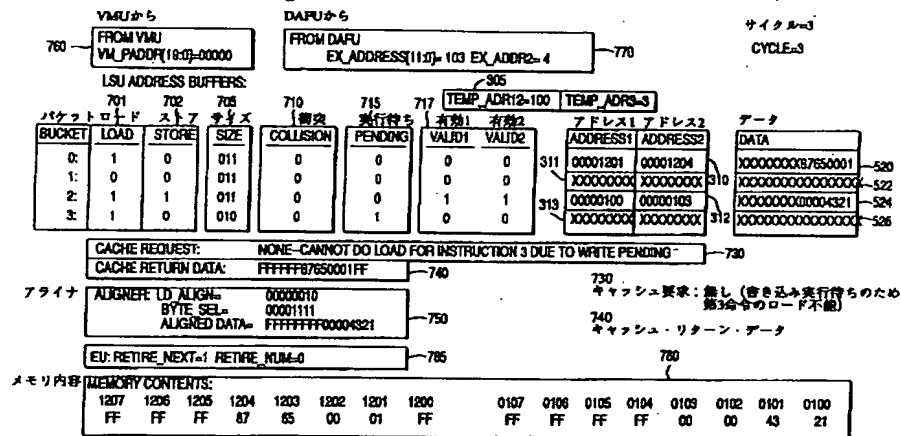


(19)

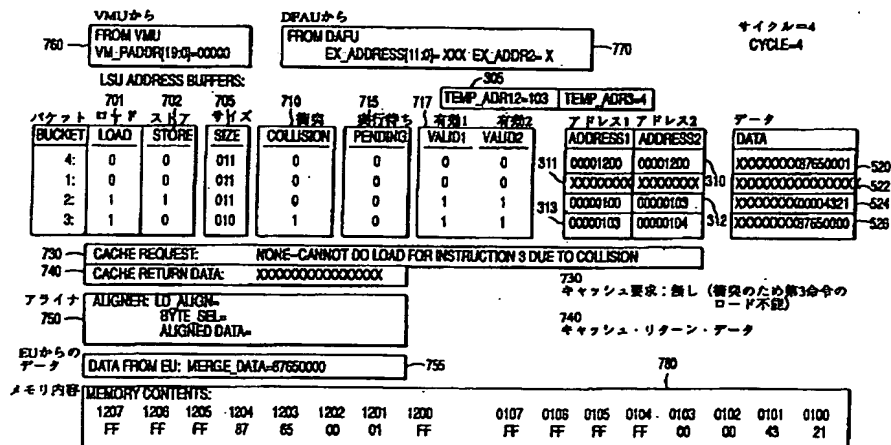
【図9】



【図10】

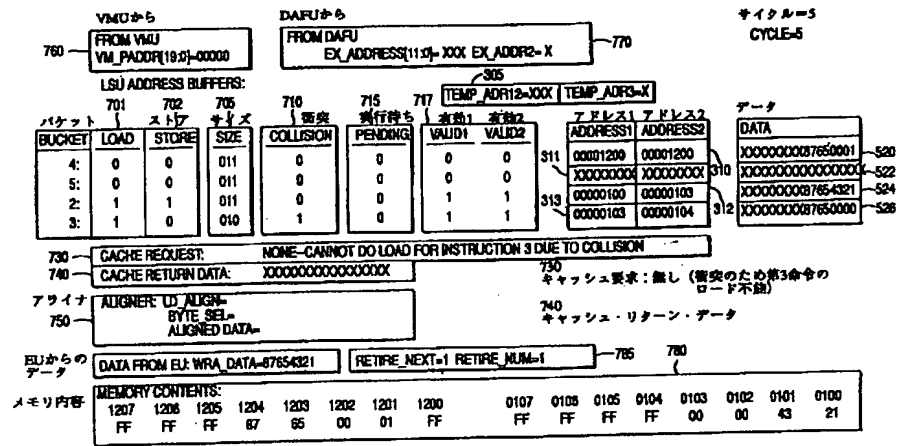


【図11】

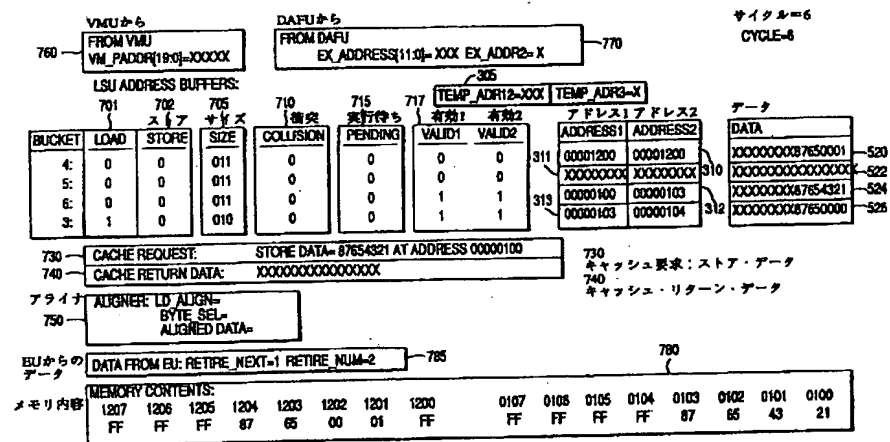


(20)

【図12】

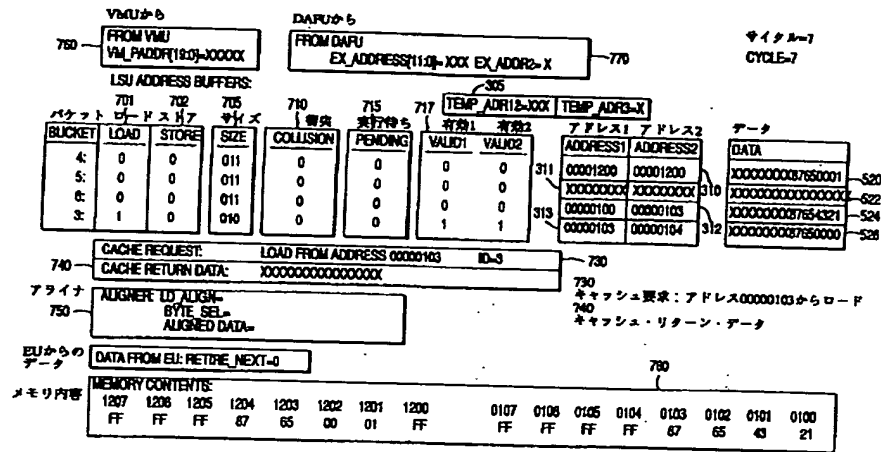


【図13】

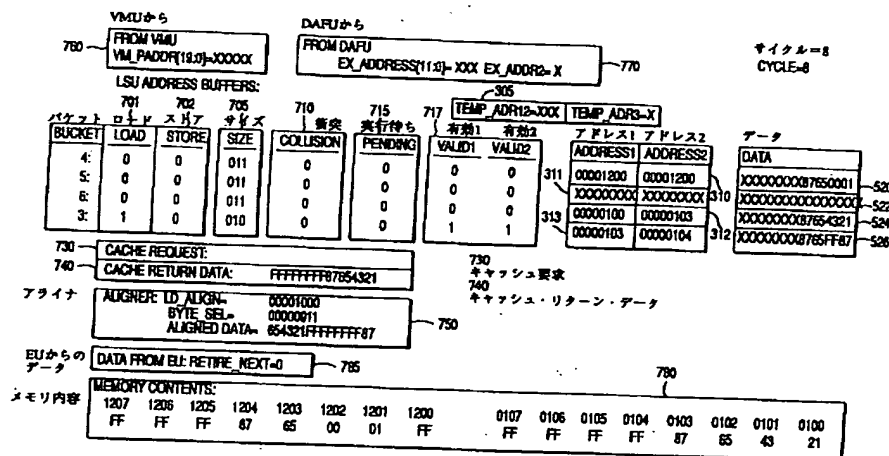


(21)

【図14】



【図15】



**THIS PAGE BLANK (USPTO)**